使用层次化可导航小世界图进行高效且稳健的近似最近邻搜索

作者：Yu. A. Malkov, D. A. Yashunin 翻译：chatgpt4o, ULis3h

摘要

我们提出了一种基于可导航小世界图的新方法，用于近似K最近邻搜索，该方法具有可控的层次结构（Hierarchial NSW， HNSW）。这个解决方案是完全基于图的，不需要任何额外的搜索结构，这通常在大多数近邻图技术的粗略搜索阶段使用， 分层NSW逐步构建一个多层结构，该结构由用于存储元素的嵌套子集的分层近邻图（层）组成。元素所在的最大层是通过指数衰减的概率分布随机选择的。这使得生成的图与之前研究的可导航小世界（NSW）结构类似，同时额外地根据其特征距离尺度分离链接。从上层开始搜索并利用尺度分离，相较于NSW提升了性能，并实现了对数复杂度的扩展。在高召回率和高度聚类数据的情况下，额外采用启发式方法来选择近似图邻域，可以显著提高性能。性能评估结果表明，所提出的一般度量空间搜索索引的性能远优于之前的开源的、最先进的纯向量方法。算法结构与跳表类似，这使得其可以直接均衡的分布式实现。

关键词

图和树的搜索策略，人工智能，信息搜索与检索，信息存储与检索，信息技术与系统，搜索过程，图和网络，数据结构，近邻搜索，大数据，近似搜索，相似性搜索。

1引言

不断增长的信息资源数量导致了对可扩展且高效的相似性搜索数据结构的高度需求。K近邻搜索（K-NNS）是信息搜索中常用的一种方法。K-NNS假设数据元素之间有一个提前定义的距离函数，目标是从数据集中找到与给定查询距离最小的K个元素。这种算法在许多应用中都有使用，例如非参数化的机器学习算法、大规模数据库中的图像特征匹配和语义文档检索。K-NNS的一种朴素方法是计算查询与数据集中每个元素之间的距离，然后选择距离最小的元素。然而，这种朴素方法的复杂度随着存储元素数量线性增长，使得它在大规模数据集上不可行。这促使人们对快速且可扩展的K-NNS算法的开发产生了极大的兴趣。

K-NNS的精确解仅在相对低维数据的情况下才能显著加快搜索速度，这是由于“维度灾难”的影响。为了解决这个问题，提出了近似最近邻搜索（K-ANNS）的概念，它通过允许少量错误来放宽精确搜索的条件。非精确搜索的质量（召回率）被定义为找到的真实最近邻数量与K的比值。最流行的K-ANNS解决方案基于树算法的近似版本、局部敏感哈希（LSH）和乘积量化（PQ）。最近，基于邻近图的K-ANNS算法因其在高维数据集上的更好表现而逐渐流行。然而，邻近图路由的幂律扩展在处理低维或聚类数据时会导致性能严重下降。

在本文中，我们提出了分层可导航小世界（Hierarchical NSW，HNSW），这是一种全新的完全基于图的增量式K-ANNS结构，能够提供更好的对数复杂度扩展。主要贡献包括：明确选择图的入口节点、通过不同的尺度分隔链接，以及使用高级启发式方法选择邻居。或者，可以将分层NSW算法视为概率跳表结构的扩展，但用邻近图替代了链表。性能评估表明，所提出的一般度量空间方法能够显著超越之前仅适用于向量空间的开源最先进方法。

2 相关研究

2.1 邻近图技术

在绝大多数研究的图算法中，搜索采用的是k近邻（k-NN）图中的贪婪路由形式。对于给定的邻近图，搜索从某个入口点开始（可以是随机的，也可以由单独的算法提供），然后迭代地遍历图。在遍历的每一步中，算法会检查查询点与当前基节点邻居之间的距离，然后选择距离最小的相邻节点作为下一个基节点，同时不断跟踪发现的最佳邻居。当满足某些停止条件（例如距离计算的次数）时，搜索终止。在k-NN图中，指向最近邻的链接作为Delaunay图（一种保证基本贪婪图遍历结果始终为最近邻的图）的简单近似。不幸的是，如果没有关于空间结构的先验信息，就无法高效地构建Delaunay图，但通过仅使用存储元素之间的距离，可以通过最近邻对其进行近似。研究表明，使用这种近似的邻近图方法在性能上与其他k-ANNS技术（如kd树或LSH）具有竞争力。

k-NN图方法的主要缺点是：
1）在路由过程中，随着数据集规模的增大，步骤数量呈幂律增长； 2）可能丧失全局连通性，从而导致在聚类数据上的搜索结果较差。为了解决这些问题，已经提出了许多混合方法，这些方法使用了仅适用于向量数据的辅助算法（例如kd树和乘积量化），通过粗略搜索为入口节点找到更好的候选点。

在文献 [25, 26, 30] 中，作者提出了一种基于邻近图的K-ANNS算法，称为可导航小世界（NSW，也称为度量化小世界，MSW）。该算法利用了可导航图，即在贪婪遍历过程中，跳数相对于网络规模呈对数或多对数增长的图。NSW图通过以随机顺序连续插入元素构建，每个新插入的元素通过双向方式与之前插入的M个最近邻连接。M个最近邻是通过该结构的搜索过程找到的（即一种从多个随机入口节点开始的贪婪搜索变体）。在构建初期插入的元素与最近邻的链接后成为网络枢纽之间的桥梁，这些桥梁保持了整体图的连通性，并允许在贪婪路由过程中跳数呈对数增长。

NSW结构的构建阶段可以在没有全局同步的情况下高效并行化，并且对准确性没有可测量的影响，因此是分布式搜索系统的一个不错选择。

NSW方法在某些数据集上实现了最先进的性能，然而，由于整体多对数复杂度的扩展，该算法在低维数据集上仍容易出现严重的性能下降（在这些数据集上，NSW可能会比基于树的算法差几个数量级）。

2.2 可导航小世界模型

具有对数或多对数扩展的贪婪图路由的网络被称为可导航小世界网络。此类网络是复杂网络理论中的一个重要研究主题，旨在理解现实生活中网络形成的潜在机制，以将其应用于可扩展路由和分布式相似性搜索。

最早研究可导航网络空间模型的工作是由 J. Kleinberg 完成的 [31, 41]，他将其作为著名的 Milgram 实验 [42] 的社会网络模型。Kleinberg 研究了随机 Watts-Strogatz 网络 [43] 的一种变体，使用 (d) 维向量空间中的规则格子图，并结合遵循特定远程链接长度分布 (r^{-\alpha}) 的远程链接增强。当 (\alpha = d) 时，通过贪婪路由到达目标的跳数呈多对数扩展（而对于其他任何 (\alpha) 值，都遵循幂律）。这一思想启发了许多基于导航效应的 K-NNS 和 K-ANNS 算法的开发 [37-40]。但即使 Kleinberg 的可导航性准则原则上可以扩展到更一般的空间中，要构建这样的可导航网络，必须事先了解数据分布。此外，Kleinberg 图中的贪婪路由在最佳情况下也会受到多对数复杂度扩展的限制。

另一类著名的可导航网络是无标度模型，它可以再现现实生活网络的若干特性，并被推荐用于路由应用。然而，由此类模型生成的网络在贪婪搜索中的复杂度扩展甚至比幂律更差，并且与 Kleinberg 模型类似，无标度模型需要数据分布的全局知识，这使得它们无法用于搜索应用。

上述描述的 NSW 算法使用了一种更简单、此前未知的可导航网络模型，该模型允许去中心化的图构建，并适用于任意空间中的数据。据建议，NSW 网络的形成机制可能是大规模生物神经网络可导航性的原因（这一点尚存争议）：类似的模型能够描述小型大脑网络的生长，而该模型预测了在大规模神经网络中观察到的若干高级特性。然而，NSW 模型在路由过程中也面临多对数搜索复杂度的问题。

3 研究动机

改进 NSW 搜索复杂性的方法可以通过对路由过程的分析来确定，该过程在文献 [32, 44] 中进行了详细研究。路由过程可以分为两个阶段：“放大”（“zoom-out”）和“缩小”（“zoom-in”）[32]。贪婪算法从“放大”阶段开始，从一个低度节点出发，同时遍历图并逐步增加节点的度，直到节点链接长度的特征半径达到与查询距离相当的尺度。在此之前，节点的平均度可能保持相对较小，这会导致陷入远处的错误局部最小值的概率增加。

通过从具有最大度数的节点（例如 NSW 结构中最早插入的节点 [44]）开始搜索，可以避免上述问题，直接进入搜索的“缩小”阶段。测试表明，将枢纽节点作为起点可以显著提高结构中成功路由的概率，并在低维数据上提供显著更好的性能。然而，即使如此，它在最佳情况下也仅具有多对数复杂度的单次贪婪搜索扩展性，并且在高维数据上的表现仍然不如分层 NSW。

NSW 中单次贪婪搜索具有多对数复杂度扩展性的原因在于，距离计算的总次数大致与贪婪算法跳数的平均值与贪婪路径上节点的平均度数的乘积成正比。跳数的平均值以对数级扩展 [26, 44]，而贪婪路径上节点的平均度数也以对数级扩展，其原因包括：
1）随着网络的增长，贪婪搜索倾向于经过相同的枢纽节点 [32, 44]；
2）随着网络规模的增加，枢纽节点连接的平均数量以对数级增长。因此，最终复杂度呈现总体上的多对数依赖关系。

分层 NSW 算法的核心思想是根据链接的长度尺度将其分为不同的层，然后在一个多层图中进行搜索。在这种情况下，对于每个元素，我们只需要评估固定比例的链接，而与网络规模无关，从而实现对数级的扩展性。在这种结构中，搜索从仅包含最长链接的上层开始（即“缩小”阶段）。算法从上层贪婪地遍历元素，直到到达局部最小值（参见图 1 以了解示意图）。之后，搜索切换到包含较短链接的下一层，从上一层的局部最小值元素重新开始，并重复这一过程。在所有层中，每个元素的最大连接数可以保持为常数，从而使得在一个可导航的小世界网络中实现路由的对数级复杂度扩展性。

构建这种分层结构的一种方法是通过引入层级，显式地设置具有不同长度尺度的链接。对于每个元素，我们选择一个整数层级 l，它定义了该元素所属的最高层级。对于每一层的所有元素，逐步构建一个邻近图（即仅包含“短”链接的图，用于近似德洛内图）。如果我们为 l 设置一个指数衰减的概率分布（即遵循几何分布），则可以实现结构中层数期望值的对数级扩展性。搜索过程是一个从最高层开始并在零层结束的迭代贪婪搜索。

l 可以与 NSW 中的节点度数对应）。与 NSW 不同的是，分层 NSW 的构建算法不需要在插入之前对元素进行打乱——通过使用层级随机化实现了随机性，从而即使在数据分布暂时发生变化的情况下，也能实现真正的增量索引（尽管改变插入顺序会稍微影响性能，因为构建过程仅部分具有确定性）。

分层 NSW 的理念与一个众所周知的一维概率跳表结构 [27] 非常相似，并可以使用跳表的术语来描述。与跳表的主要区别在于，我们通过将链表替换为邻近图来对结构进行了泛化。因此，分层 NSW 方法可以利用相同的方法来构建分布式近似搜索/覆盖结构 [45]。

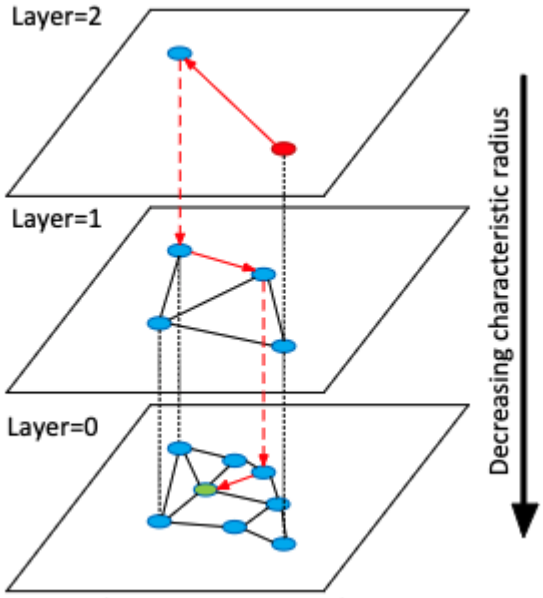

图 1. 分层 NSW 思想的示意图。搜索从顶层的一个元素开始（红色显示）。红色箭头表示从入口点到查询点（绿色显示）的贪婪算法方向。

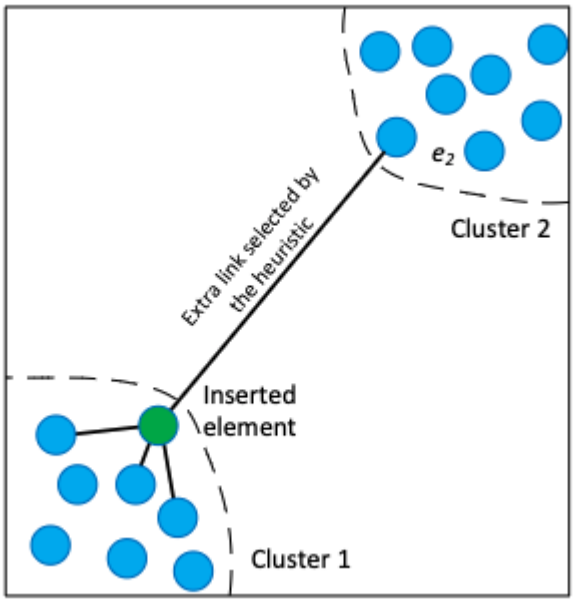

图 2. 用于选择两个孤立聚类的近邻图的启发式方法示意图。一个新元素被插入到聚类 1 的边界上。该元素的所有最近邻都属于聚类 1，因此会错过聚类之间德洛内图的边。然而，该启发式方法选择了来自聚类 2 的元素 e2，从而在插入的元素相比聚类 1 中的其他任何元素更接近 e2 的情况下，保持了全局连通性。

在元素插入过程中，为选择邻近图的连接，我们使用了一种启发式方法，该方法考虑候选元素之间的距离，以创建多样化的连接（类似的算法曾在空间近似树 [4] 中用于选择树的子节点），而不仅仅是选择最近的邻居。该启发式方法从最接近插入元素的候选开始检查，并仅在候选元素相对于基准（插入的元素）比任何已连接的候选元素更近时，才创建与该候选的连接（详见第 4 节）。

当候选元素的数量足够多时，该启发式方法可以得到精确的相对邻域图 [46] 作为子图，这是德洛内图的一个最小子图，仅通过节点之间的距离即可推导出。相对邻域图即使在数据高度聚类的情况下，也能轻松保持全局连通组件（见图 2 示例）。需要注意的是，与精确的相对邻域图相比，该启发式方法会创建额外的边，从而可以控制连接的数量，这对于搜索性能至关重要。在一维数据的情况下，该启发式方法仅通过元素之间的距离信息即可获得精确的德洛内子图（在这种情况下与相对邻域图一致），从而实现从分层 NSW 到一维概率跳表算法的直接过渡。

分层 NSW 邻近图的基础变体也在文献 [18] 中用于邻近图搜索（称为“稀疏邻域图”）。类似的启发式方法也是 FANNG 算法 [47] 的研究重点（该算法在当前论文初版发布在线后不久发表），但其解释稍有不同，基于稀疏邻域图的精确路由特性 [18]。

算法 1

INSERT(hnsw, q, M, Mmax, efConstruction, mL)

输入:

- **hnsw**: 多层图
- **q**: 新元素
- **M**: 已建立的连接数
- **Mmax**: 每层每个元素的最大连接数
- **efConstruction**: 动态候选列表的大小
- **mL**: 层级生成的归一化因子

输出:
更新 hnsw，插入元素 (q)

---

1. (W \leftarrow \emptyset) // 当前找到的最近元素列表
2. (ep \leftarrow) 获取 hnsw 的入口点
3. (L \leftarrow) (ep) 的层级 // hnsw 的顶层
4. (l \leftarrow \lfloor -\ln(\text{unif}(0..1)) \cdot mL \rfloor) // 新元素的层级
5. **for** (lc \leftarrow L \dots l+1):
   6. (W \leftarrow \text{SEARCH-LAYER}(q, ep, ef=1, lc))
   7. (ep \leftarrow) 从 (W) 中获取最接近 (q) 的元素
6. **for** (lc \leftarrow \min(L, l) \dots 0):
   9. (W \leftarrow \text{SEARCH-LAYER}(q, ep, efConstruction, lc))
   10. (neighbors \leftarrow \text{SELECT-NEIGHBORS}(q, W, M, lc)) // 使用算法 3 或算法 4
   11. 在层 (lc) 双向添加从 (neighbors) 到 (q) 的连接
   12. **for each** (e \in neighbors): // 如果需要，缩减连接
   13. (eConn \leftarrow) (e) 在层 (lc) 的邻域
   14. **if** (|eConn| > Mmax): // 缩减 (e) 的连接
   // 如果 (lc = 0)，则 (Mmax = Mmax0)
   15. (eNewConn \leftarrow \text{SELECT-NEIGHBORS}(e, eConn, Mmax, lc)) // 使用算法 3 或算法 4
   16. 将 (e) 在层 (lc) 的邻域设置为 (eNewConn)
   17. (ep \leftarrow W)
7. **if** (l > L):
   19. 将 hnsw 的入口点设置为 (q)